MUNICH CYBER TACTICS
TECHNIQUES AND PROCEDURES | 25

MCTTP

COM
to the
DARKSIDE

Speakers

Dylan Tran (@d_tranman)

- Red Team Consultant @ IBM X-Force as of 1 month ago
- Former D1 Hooper turned sillyware developer
- Pursuer of flavors of liquid origin

Jimmy Bayne (@bohops)

- Red Team Consultant @ IBM X-Force
- Still waiting for a time machine destined for ’96
- Possibly older than COM

Agenda

A long time ago in the 90s far, far away…

Introduction to COM

Cross Session Attacks

Case Study: Windows Security Health Agent

Lateral Movement

Case Study: Trapped COM Objects

Defensive Considerations

…and some other interesting things thrown in the mix

Disclaimer: Any views or opinions expressed in this presentation are our own and do not necessarily reflect those of our employer

Motivation for COM Research

- Historical attack surface
- Capability enhancements
- Defense evasion
- *Uncanny obsession that may require professional help one day…*

Source: https://i.redd.it/i2jml7czupg81.jpg

Introduction to COM

Source: https://historyuk.s3.eu-west-2.amazonaws.com/s3fs-public/styles/768x432/public/2020-07/ancient-aliens-1.png?itok=WeJUQRjn

Introduction to COM

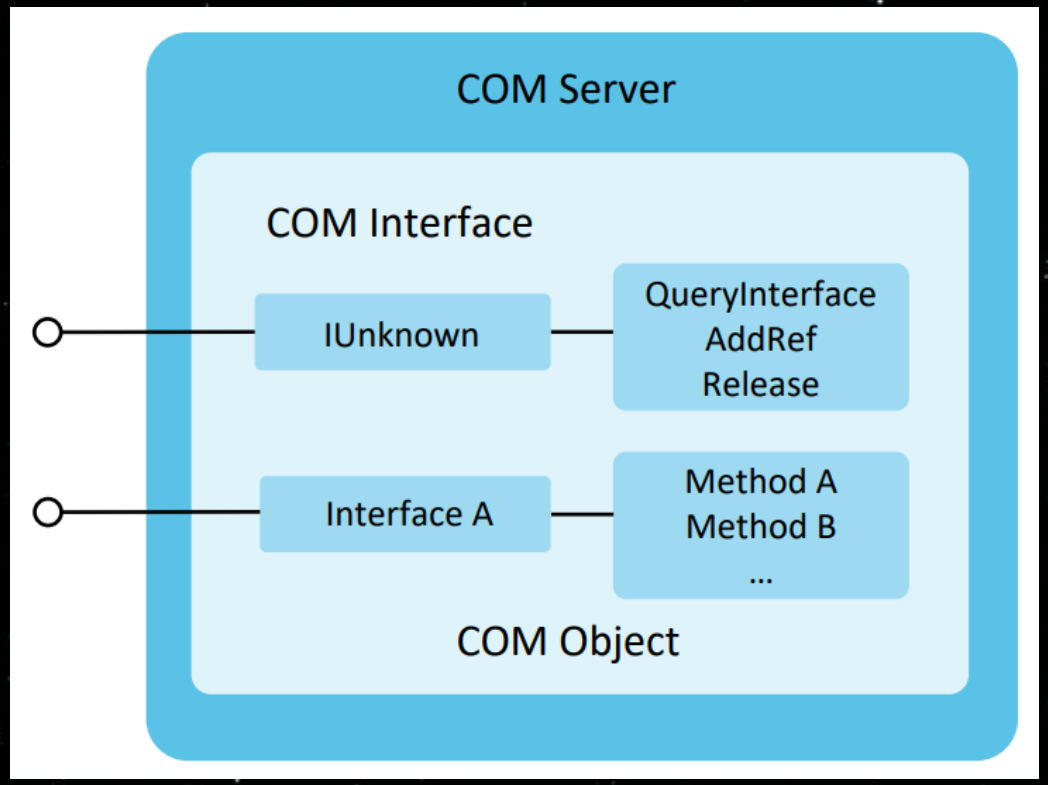
COM: Component Object Model

- Application Binary Interface (ABI) for ‘agnostic’ components
- Essentially a middleware allowing disparate components to communicate
- Common client/server models:
  - Local process on the same system (“In Process”)
  - Remote process on the same system (“Out of Process”)

DCOM: Distributed Component Object Model

- In practice, extends standard COM to enable communication between components across process and/or machine boundaries via configurable infrastructure for controlling activation, security, and communication policies
- Common client/server models:
  - Remote process on the same system (via ALPC channel)
  - Remote process on a different system (via RPC channel)

Introduction to COM: (Just) COM



Source: **CertifiedDCOM**
https://i.blackhat.com/Asia-24/Presentations/Asia-24-Ding-CertifiedDCOM-The-Privilege-Escalation-Journey-to-Domain-Admin.pdf

Terminology	Description
Server	The executable code that provides the implementation of one or more COM classes.
Client	COM consumer that accesses services via COM objects and implemented interfaces.
Class	A concrete implementation of a group of interfaces within a COM server.
Object	An activated instance of a COM class
Interface	A “contract” that defines a collection of methods implemented by a COM Object. Interfaces use **vtables** (function pointer tables) for dispatching calls to the implemented methods.

Introduction to COM: Registry

Registered COM

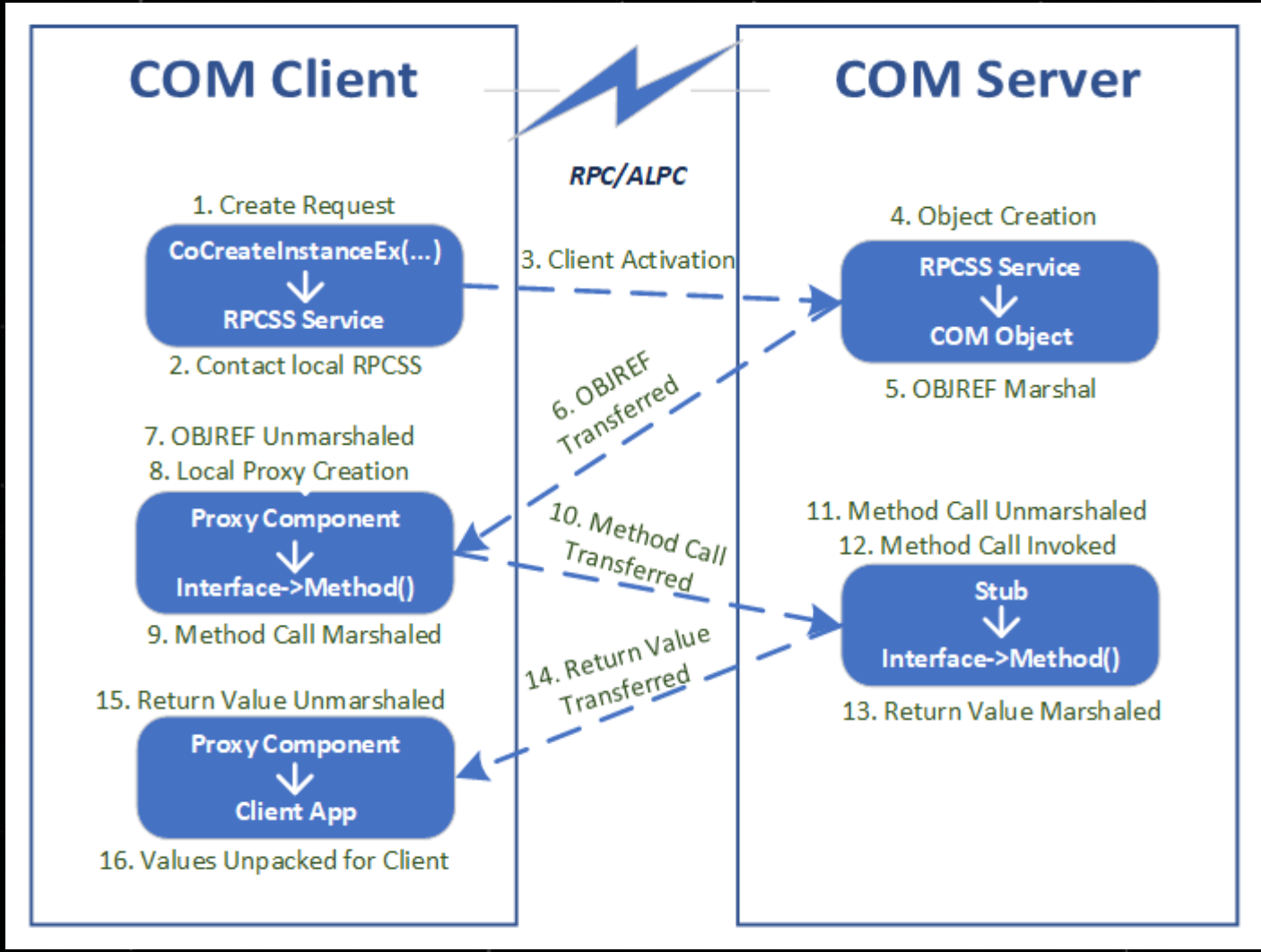
• COM Class Settings
  • Location: *HKCR\CLSID\{GUID}*
  • Servers: InprocServer32, LocalServer32
  • DCOM Association: AppID value

• DCOM Configuration Settings (AppID)
  • Location: *HKCR\AppID\{GUID}*
  • Security Settings: Launch and Access permissions
  • Default System-wide Security Settings: *HKLM\SOFTWARE\Microsoft\Ole*
  • Identity Settings: Run-As user (e.g. Interactive User, Service, System)
  • Surrogate Settings: DCOM libraries running in a process (e.g. dllhost.exe)

• Interface Definitions   • Location: *HKCR\Interface\{GUID}*

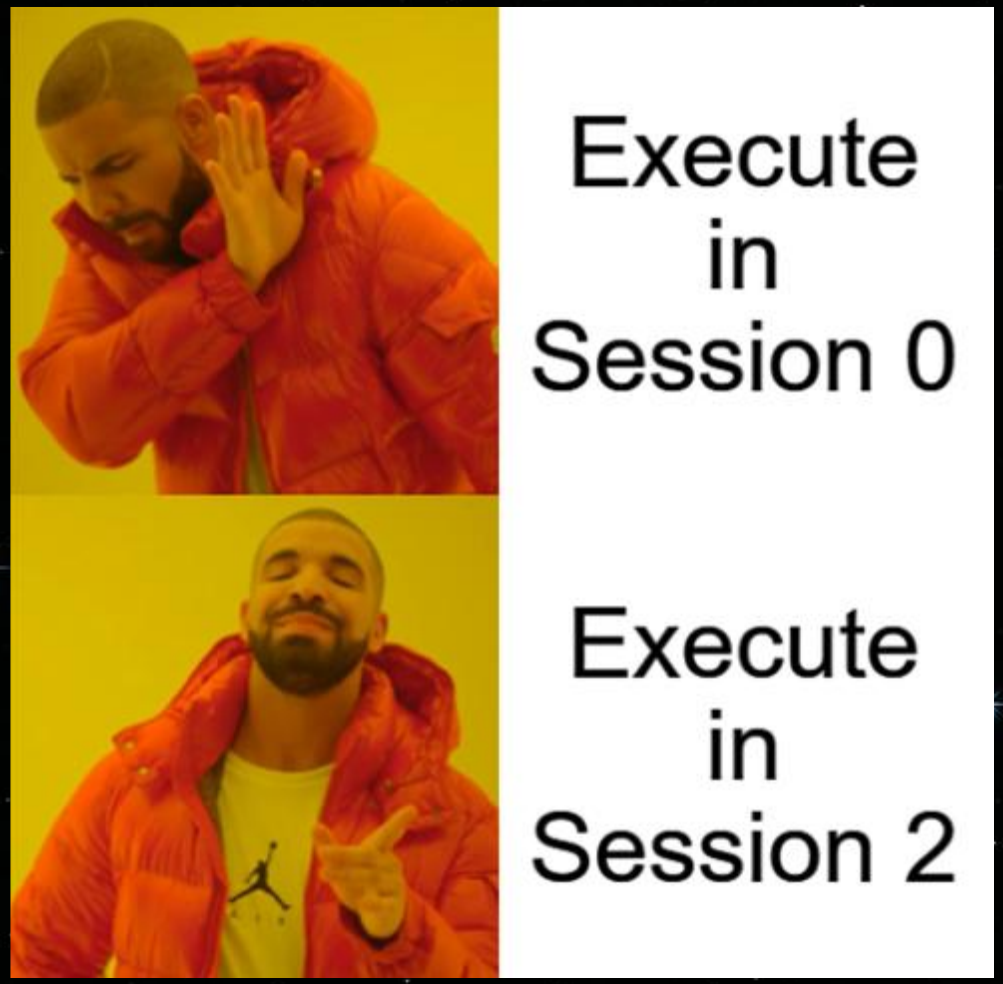
• Type Libraries
  • Defined: Metadata that describes COM object interfaces, methods, and other type information
  • Location: *HKCR\TypeLib\{GUID}*

Introduction to COM: DCOM



Terminology	Description
Marshal	Packaging method calls, parameters, and data into a format that can be transmitted across a process or machine boundary.
Proxy	A stand-in component that exposes the same Interface in the COM client-side process that marshals calls, parameters, and data into a format for transport to the DCOM server stub.
Stub	A DCOM server-side component that receives the marshaled calls, parameters, and data, performs unmarshing, and calls the methods for the real COM object. The stub is also responsible for packaging return values and structures (e.g. **OBJREF**) back to the client.

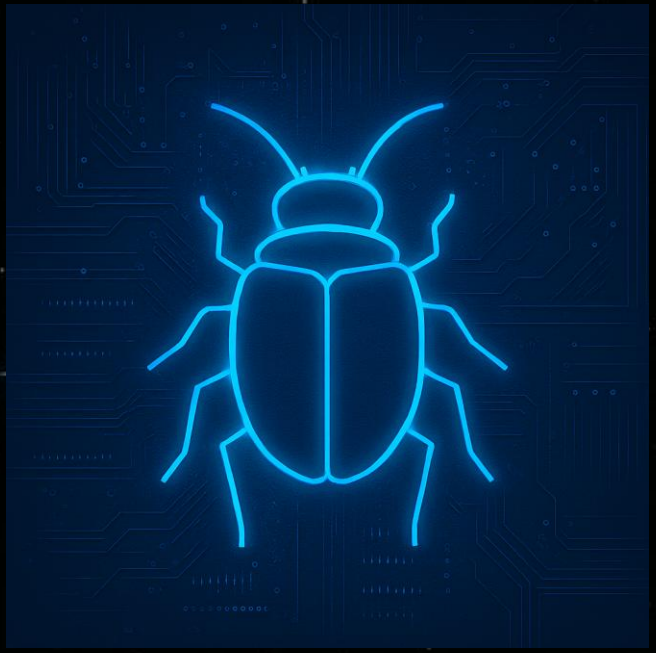
Cross Session Attacks



Source: https://imgflip.com/memegenerator/Drake-Hotline-Bling

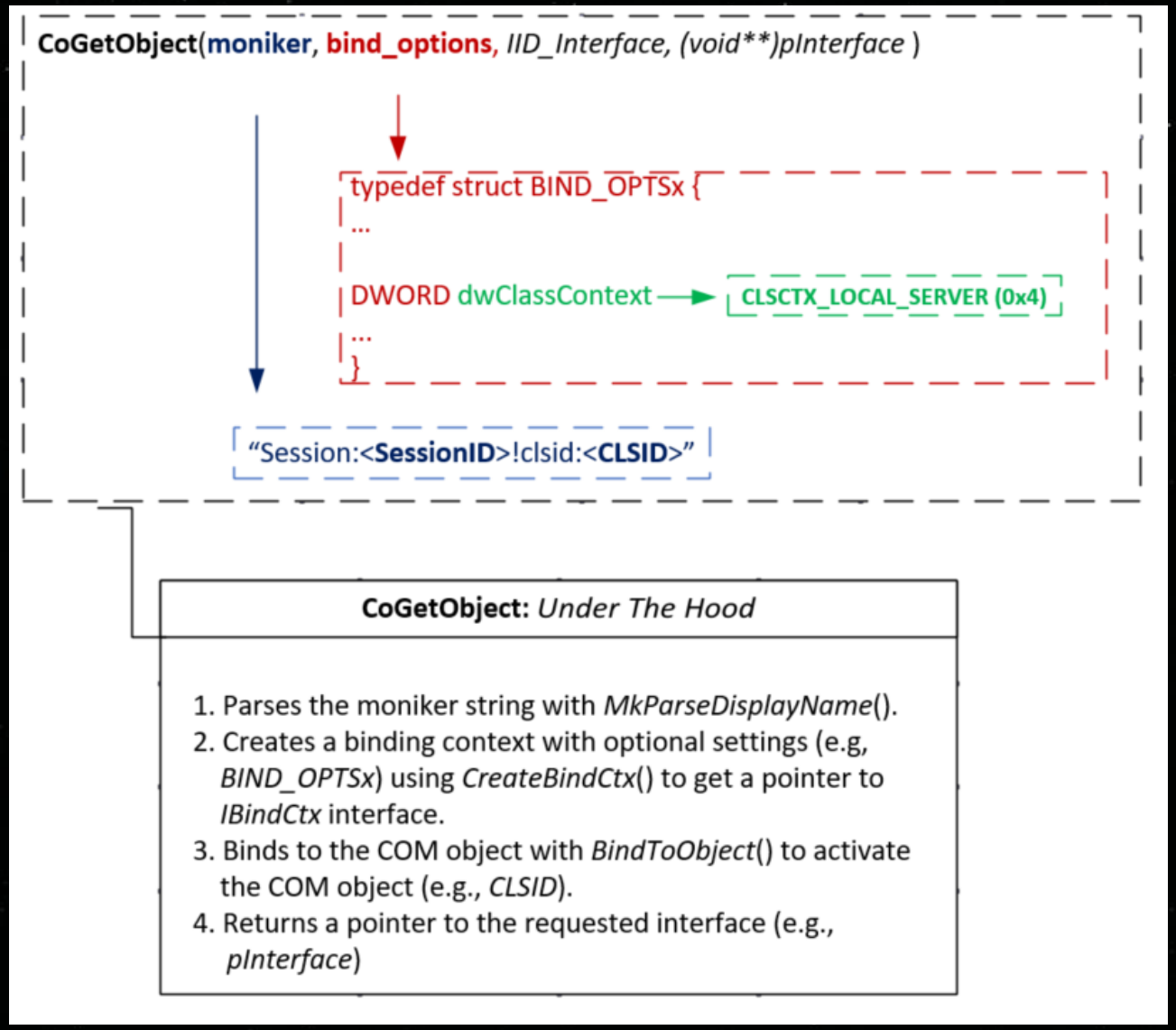
Cross Session Attack Surface

- Numerous COM objects can be ***activated*** and ***accessed*** in another user's session on the same host
  - General Use Case: Unprivileged user executes a DCOM server executable in privileged user session
- An interesting attack class for:
  - Privilege Escalation
  - Impersonation
  - Spoofing/Relays
- Vulnerability discovery opportunities with:
  - Interface method and property inspection
  - Fuzzing
  - Weird Logic Bugs



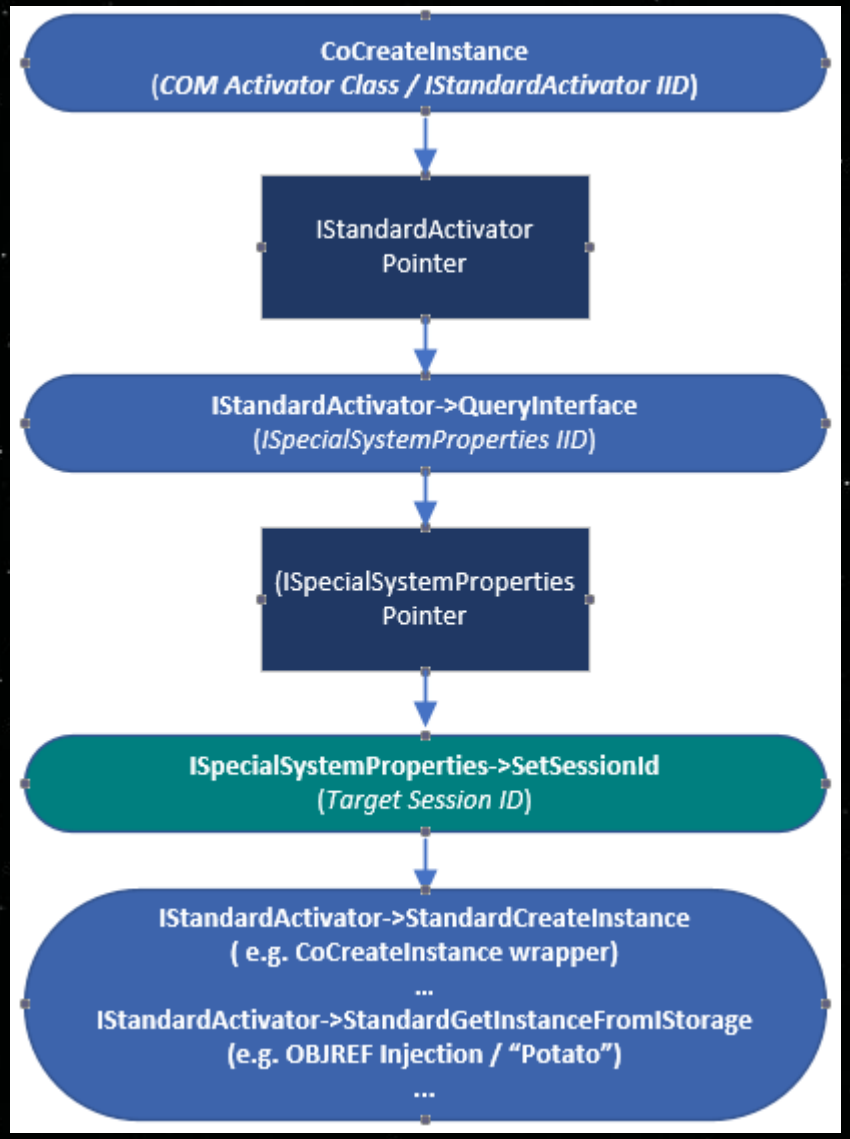
Cross Session Activation: COM Moniker

- ‘Documented’ procedure for activating cross-session COM objects with:
  - CoGetObject()
  - Marshal.BindToMoniker() (.NET)
- *Session ID specified in the moniker string*



Cross Session Activation: Standard Activator

- ‘Undocumented’ procedure for activating cross-session COM objects with:
  - IStandardActivator
  - ISpecialSystemProperties
- Specify Session ID in the *ISpecialSystemProperties->SetSessionId()* method
- Reference: https://www.tiraniddo.dev/2021/04/standard-activating-yourself-to.html



Cross Session Spoof|Relay|Potato Resources

Tool/Technique	Author(s)	Reference
RemotePotato0	Andrea Pierini Antonio Cocomazzi	https://github.com/antonioCoco/RemotePotato0
SilverPotato	Andrea Pierini	https://decoder.cloud/2024/04/24/hello-im-your-domain-admin-and-i-want-to-authenticate-against-you/
KrbRelay	Cube0x0 James Forshaw (inspiration)	https://github.com/cube0x0/KrbRelay
RemoteKrbRelay	Michael Zhmaylo	https://github.com/CICADA8-Research/RemoteKrbRelay

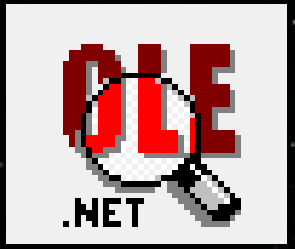
Tool/Technique	Author(s)	Reference
CertifiedDCOM	Tianze Ding	https://i.blackhat.com/Asia-24/Presentations/Asia-24-Ding-CertifiedDCOM-The-Privilege-Escalation-Journey-to-Domain-Admin.pdf
Windows Is and Always Will be a Potatoland	Nico Viakowski	https://www.r-tec.net/r-tec-blog-windows-is-and-always-will-be-a-potatoland.html
Revisiting Cross Session Activation Attacks	Fabian Mosch	https://www.r-tec.net/r-tec-blog-revisiting-cross-session-activation-attacks.html

Cross Session EoP Resources

Tool/Technique	Author(s)	Reference
CVE-2017-0100: EoP in HxHelperPaneServer	James Forshaw	https://project-zero.issues.chromium.org/issues/42450026
CVE-2019-0566: EoP in the Microsoft Edge Browser Broker	James Forshaw	https://project-zero.issues.chromium.org/issues/42450750
CVE-2021-23874/23875: EoP in McAfee CoManageOem/ManagerOem	Denis Skvortcov	https://the-deniss.github.io/posts/2021/05/17/discovering-and-exploiting-mcafee-com-objects.html
CVE-2023-33127: EoP in .NET (PhoneExperienceHost)	Jimmy Bayne	https://bohops.com/2023/11/27/abusing-net-core-clr-diagnostic-features-cve-2023-33127/
CVE-2024-38100: EoP in Explorer ShellWindows	Andrea Pierini	https://decoder.cloud/2024/08/02/the-fake-potato
CVE-2024-7023: EoP in Chrome Updater	Sylvain Heiniger	https://blog.compass-security.com/2024/10/com-cross-session-activation

Cross-Session Discovery Methodology

- General Methodology
  1. Identify cross-session COM objects
  2. Create a test harness
  3. Execute the test case and observe results
- Tools of the Trade
  - OleViewDotNet by James Forshaw
  - Disassembler (e.g. IDA, Ghidra, etc.)



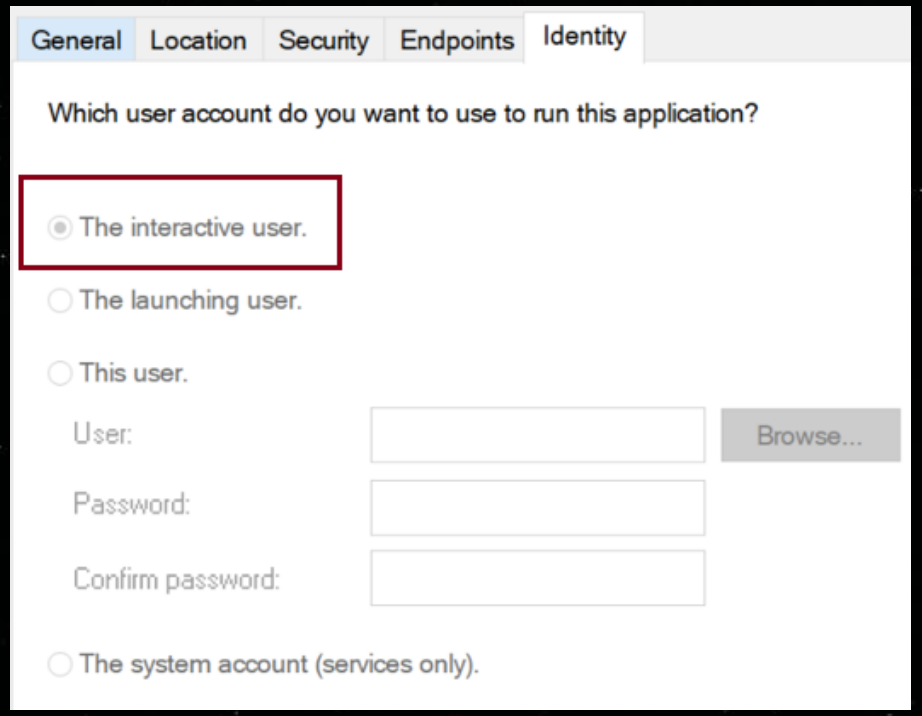
Sources
OleViewDotNet: https://github.com/tyranid/oleviewdotnet
Ghidra: https://github.com/NationalSecurityAgency/ghidra
IDA: https://hex-rays.com/ida-pro

Identifying Cross-Session COM Objects

APPID Setting	Description
RunAs	Interactive User
LaunchPermission	Allow account to activate the COM object
AccessPermission	Allow account to interact with the COM object

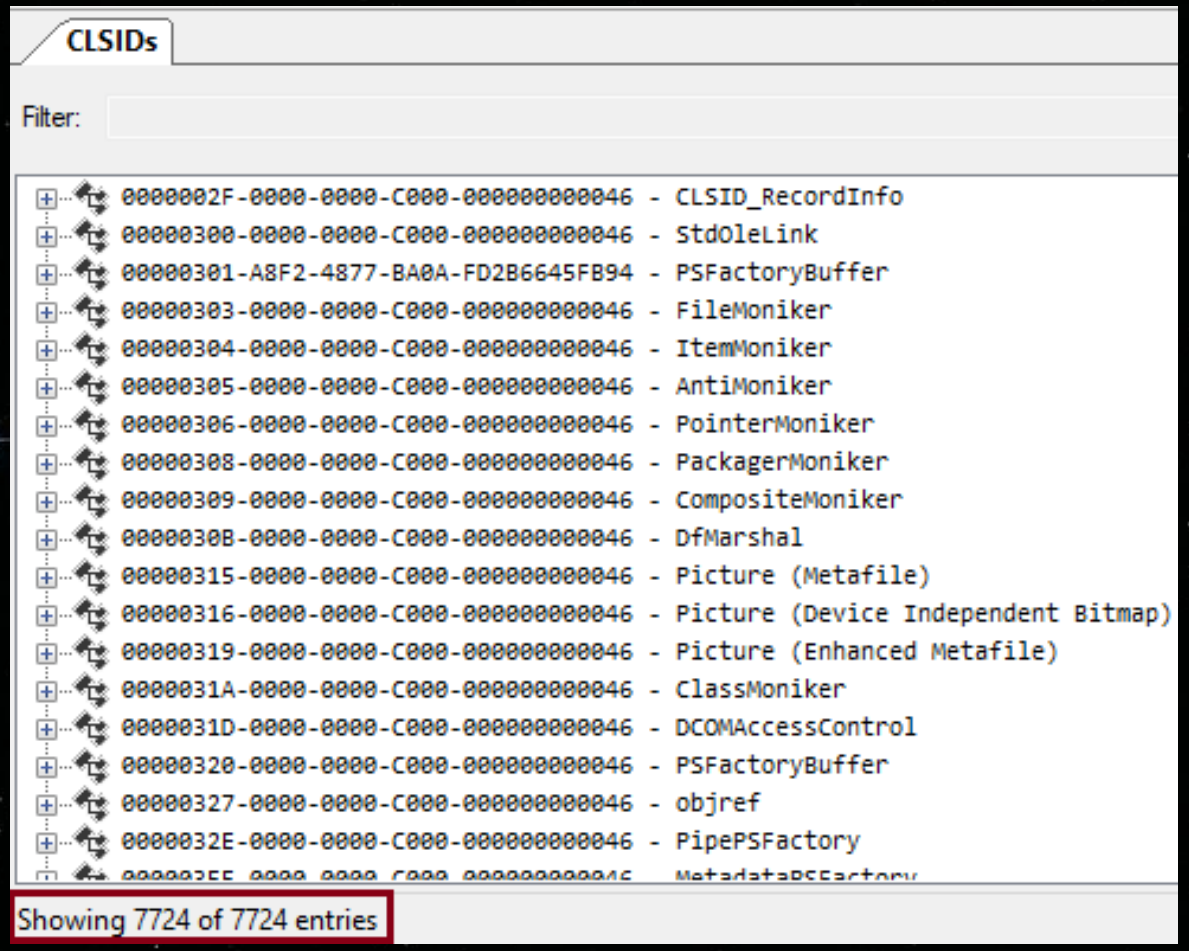
Computer\HKEY_CLASSES_ROOT\AppID\{1D278EEF-5C38-4F2A-8C7D-D5C13B662567}

Name	Type	Data
(Default)	REG_SZ	Security Health Agent Host
AccessPermission	REG_BINARY	01 00 14 80 a8 00 00 00 b8 00 00 00 14 00 00 00 30 00 00 ...
AppIDFlags	REG_DWORD	0x00000208 (520)
DllSurrogate	REG_SZ	\\?\C:\Windows\System32\SecurityHealthHost.exe
LaunchPermission	REG_BINARY	01 00 14 80 bc 00 00 00 cc 00 00 00 14 00 00 00 30 00 00 ...
RunAs	REG_SZ	Interactive User



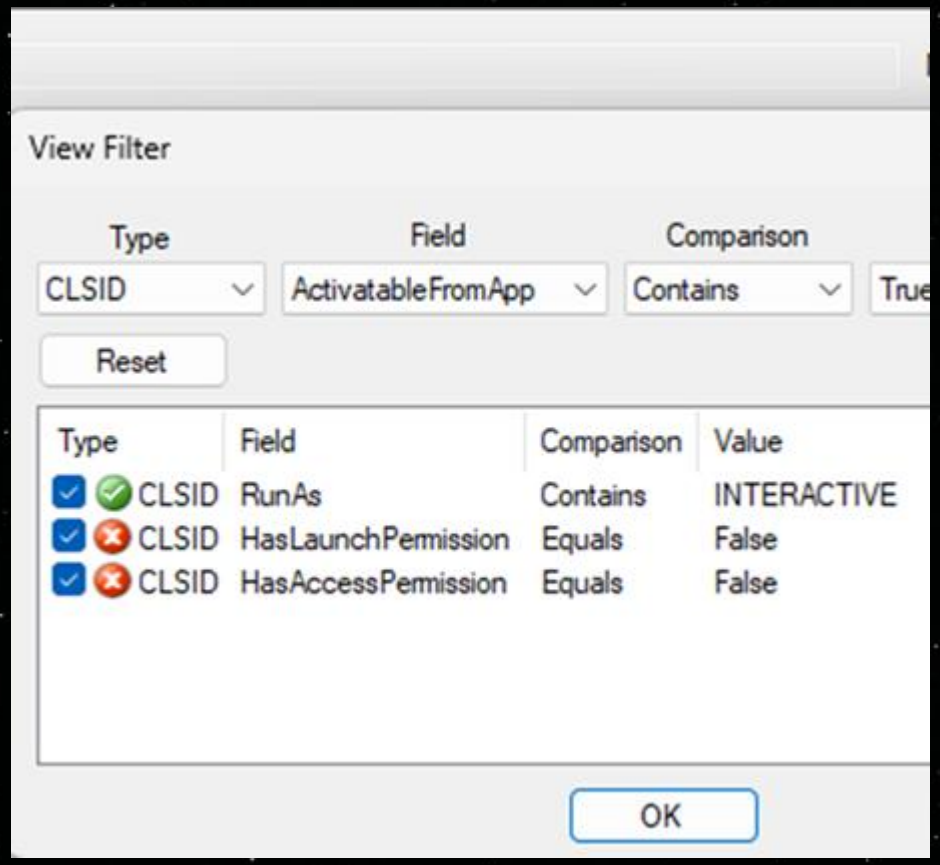
Identifying Cross-Session COM Objects

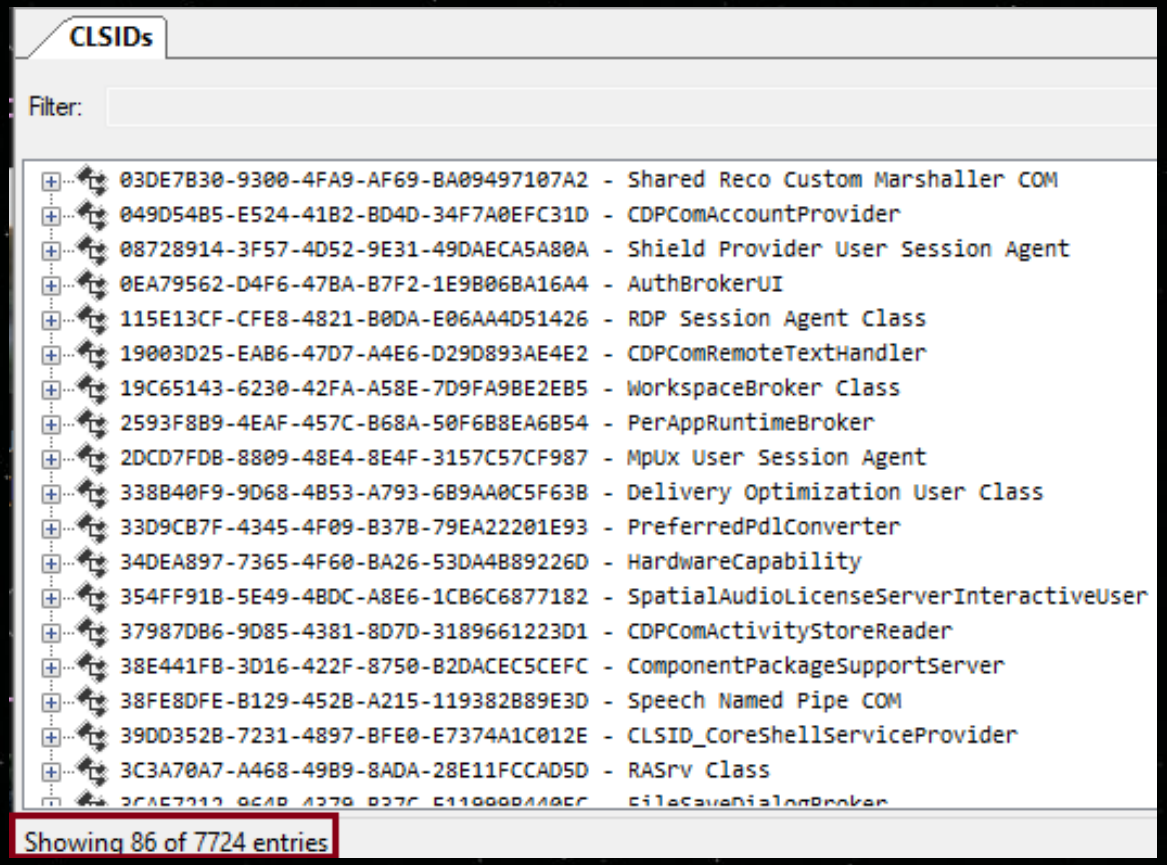
- OleViewDotNet: List all Classes/CLSIDs



Inspecting Cross-Session COM Objects

- OleViewDotNet: Apply a “complex” filter





Inspecting Cross-Session COM Objects

❖Launch Permissions

Owner: BUILTIN\Administrators
Group: BUILTIN\Administrators
Integrity: Low (NoExecuteUp)

DACL SACL

Flags: None

ACL Entries

Type	Account	Access	Flags
Allowed	NT AUTHORITY\SYSTEM	GenericAll	None
Allowed	BUILTIN\Administrators	GenericAll	None
Allowed	NT AUTHORITY\INTERACTIVE	GenericAll	None
Allowed	NT AUTHORITY\LOCAL SERVICE	GenericAll	None
Allowed	microsoft.sechealthui_8wekyb3d8bbwe	Execute|ExecuteLocal|ActivateLocal	None

❖Access Permissions

Owner: BUILTIN\Administrators
Group: BUILTIN\Administrators
Integrity: Low (NoExecuteUp)

DACL SACL

Flags: None

ACL Entries

Type	Account	Access	Flags
Allowed	NT AUTHORITY\SYSTEM	Execute|ExecuteLocal	None
Allowed	BUILTIN\Administrators	Execute|ExecuteLocal	None
Allowed	NT AUTHORITY\INTERACTIVE	Execute|ExecuteLocal	None
Allowed	microsoft.sechealthui_8wekyb3d8bbwe	Execute|ExecuteLocal	None

Inspecting Cross-Session COM Objects

- Out-of-Process Server (LocalServer32)

```
38E441FB-3D16-422F-8750-B2DACEC5CEFC - ComponentPackageSupportServer
38FE8DFE-B129-452B-A215-119382B89E3D - S
39DD352B-7231-4897-BFE0-E7374A1C012E - C
3C3A70A7-A468-49B9-8ADA-28E11FCCAD5D - R
3CAE7212-964B-4379-B37C-E11999B4405C - F
49010C18-B110-421A-9047-ADCA421CBC40 - IMRequestRA Class
```

```
CLSID: 38E441FB-3D16-422F-8750-B2DACEC5CEFC
Name: ComponentPackageSupportServer
LocalServer32: C:\Windows\System32\CompPkgSrv.exe
AppID: 38E441FB-3D16-422F-8750-B2DACEC5CEFC
```

- In-Process Server (InProcServer32)

```
E041C90B-68BA-42C9-991E-477B73A75C90 - Shield Broker
E476E4C0-409C-43CD-BBC0-5905B4138494 - Hardware Shield Broker
EDAE4045-CAE6-4706-8973-FA69715B8C10 - H
F6976CF5-68A8-436C-975A-40BE53616D59 - D
```

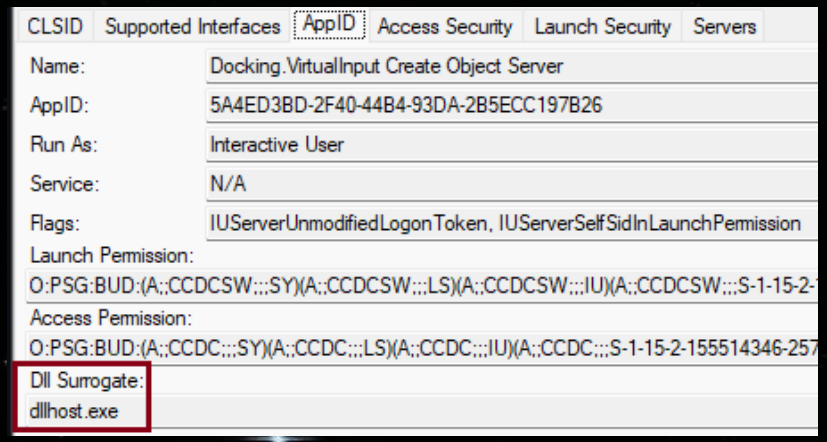
```
CLSID: E041C90B-68BA-42C9-991E-477B73A75C90
Name: Shield Broker
InProcServer32: \\?\C:\Windows\System32\SecurityHealthAgent.dll
AppID: 1D278EEF-5C38-4F2A-8C7D-D5C13B662567
Instance Interfaces: 2
Factory Interfaces: 2
```

- Out-of-Process In-Process Server (DLL Surrogate)

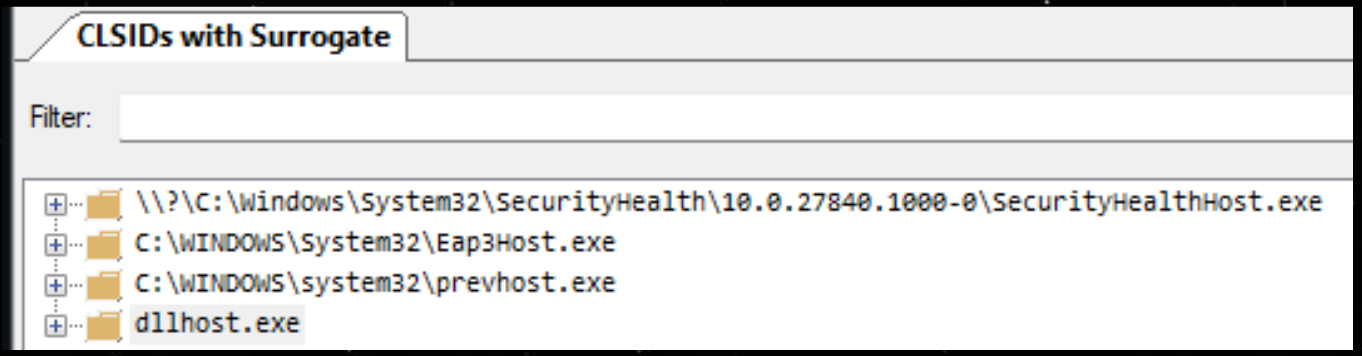
Source: https://i.pinimg.com/736x/5a/d6/ca/5ad6ca95101947299681dfa46838a1db--austin-powers-tv-quotes.jpg

Inspecting Cross-Session COM Objects

- Standard DLL Surrogate – Dllhost.exe

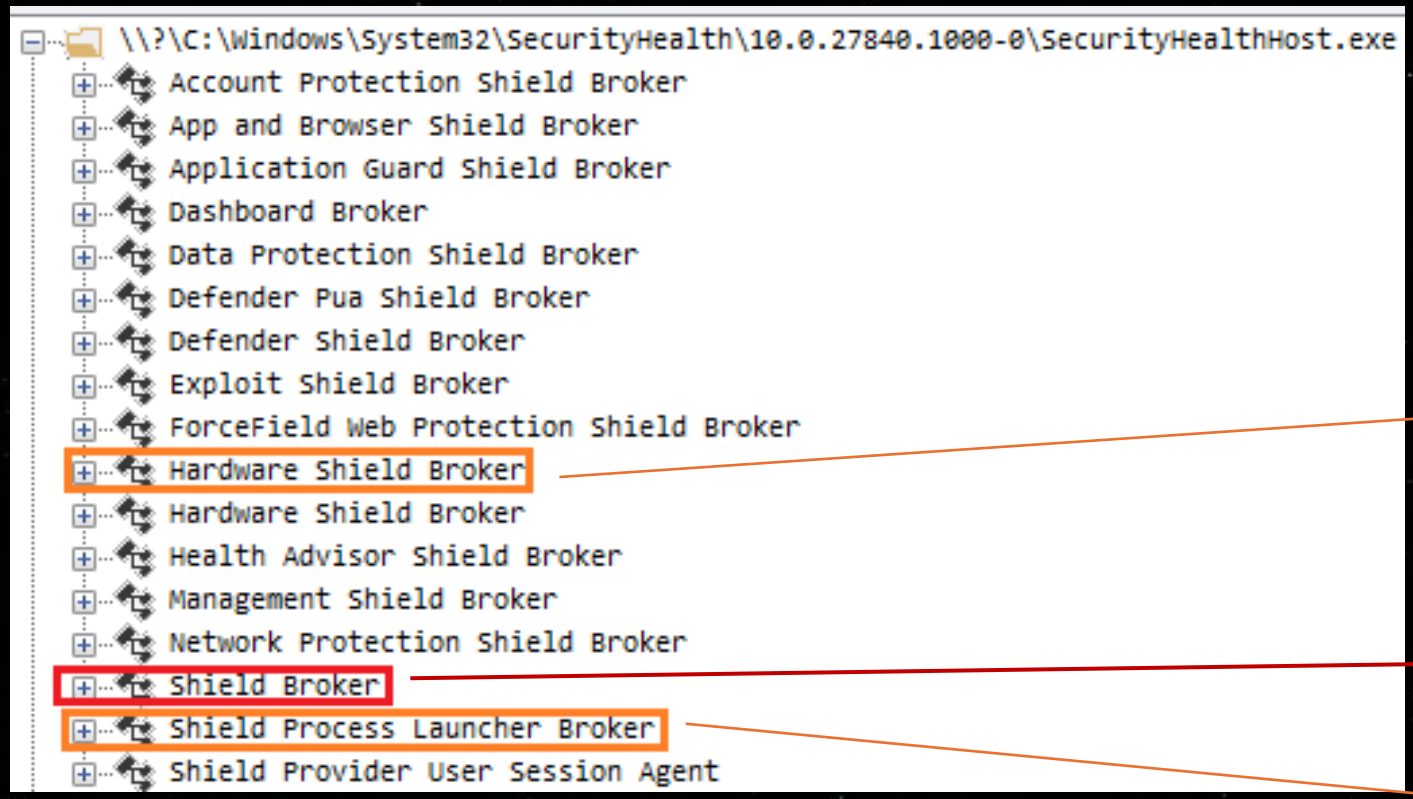


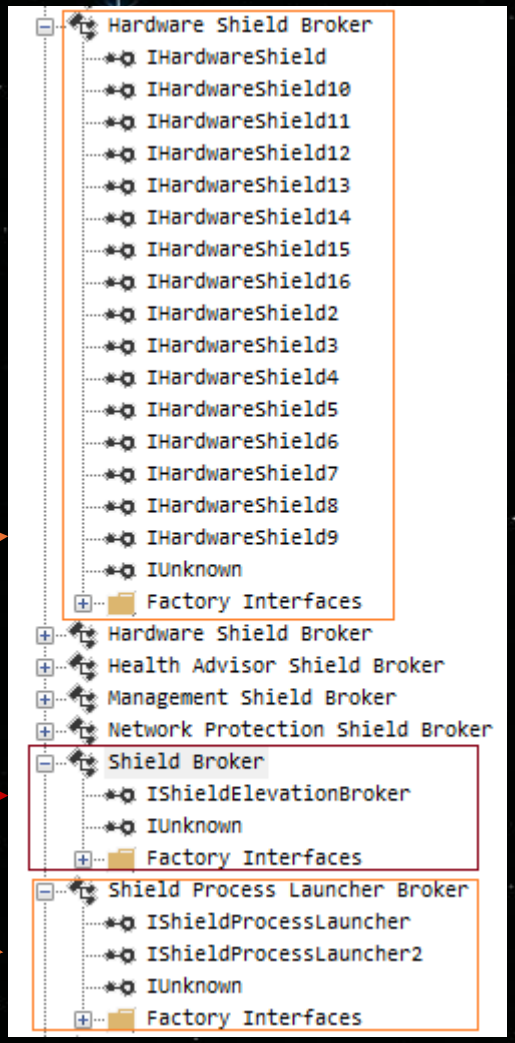
- Other Surrogates



Inspecting Cross-Session COM Objects

- Security Health Agent COM Classes





Inspecting Cross-Session COM Objects

- Security Health Agent RunAs Settings

CLSIDs with Surrogate | **Hardware Shield Broker Prope...**

CLSID | Supported Interfaces | AppID | Access Security | Launch Security | Servers | Elevation

Name:	Security Health Agent Host
AppID:	37096FBE-2F09-4FF6-8507-C6E4E1179839
Run As:	N/A
Service:	N/A
Flags:	IUServerUnmodifiedLogonToken

Shield Process Launcher Broke...

CLSID | Supported Interfaces | AppID | Access Security | Launch Security | Servers | Elevation

Name:	Security Health Agent Activate As Activator Host
AppID:	37096FBE-2F09-4FF6-8507-C6E4E1179893
Run As:	N/A
Service:	N/A
Flags:	IUServerUnmodifiedLogonToken

CLSIDs with Surrogate | **Shield Broker Properties**

CLSID | Supported Interfaces | AppID | Access Security | Launch Security | Servers

Name:	Security Health Agent Host
AppID:	1D278EEF-5C38-4F2A-8C7D-D5C13B662567
Run As:	Interactive User
Service:	N/A
Flags:	IUServerUnmodifiedLogonToken, IUServerUnmodifiedClientLogonTokenUser

Inspecting Cross-Session COM Objects

- OleViewDotNet attempts to reconstructing interface metadata information. Where a Type Library or revealing “symbol” metadata source is unavailable, generated placeholder names are used.

```
]
interface IShieldElevationBroker : IUnknown {
    HRESULT Proc3([in] GUID* p0, [in] GUID* p1, [out, iid_is(p1)]  IUnknown** p2);
    HRESULT Proc4([in] GUID* p0, [in] GUID* p1, [out, iid_is(p1)]  IUnknown** p2);
}
```

- When generic placeholders are used, leverage a disassembler to retrieve the PDB and reveal symbols and names of the implementation.

```
ShieldElevationBroker::CoCreateInstanceAsAdmin(_GUID const &,_GUID const &,void * *)
ShieldElevationBroker::CoCreateInstanceAsUser(_GUID const &,_GUID const &,void * *)
```

Inspecting Cross-Session COM Objects


```
__int64 __fastcall ShieldElevationBroker::CoCreateInstanceAsUser(
        ShieldElevationBroker *this,
        const struct _GUID *a2,
        const struct _GUID *a3,
        void **a4)
{
  HRESULT Instance; // ebx
  CommonUtil *v8; // rcx
  __int64 v9; // rdx

  Instance = ShieldElevationBroker::ValidateClsidInNormalAllowList(this, a2);
  if ( Instance < 0 )
  {
    v8 = WPP_GLOBAL_Control;
    if ( WPP_GLOBAL_Control == (CommonUtil *)&WPP_GLOBAL_Control || (*((_BYTE *)
      return (unsigned int)Instance;
    v9 = 15LL;
LABEL_5:
    WPP_SF_D_0(*((_QWORD *)v8 + 2), v9, &WPP_ab50c646a036314775501782a7272260_Tr
    return (unsigned int)Instance;
  }
  Instance = CoCreateInstance(a2, 0LL, 3u, a3, a4);
  if ( Instance < 0 )
  {
```

Inspecting Cross-Session COM Objects

- CLSID Activation Allow List

```
v11[0] = GUID_06622d85_6856_4460_8de1_a81921b41c4b;
v11[1] = CLSID_DefenderShieldBroker;
v11[2] = CLSID_DefenderPuaShieldBroker;
v11[3] = CLSID_NetworkProtectionShieldBroker;
v11[4] = CLSID_ShieldProcessLauncherBroker;
v11[5] = CLSID_AppAndBrowserShieldBroker;
v11[6] = CLSID_DashboardBroker;
v11[7] = CLSID_ExploitShieldBroker;
v11[8] = CLSID_AccountProtectionShieldBroker;
v11[9] = CLSID_HardwareShieldBroker;
v11[10] = CLSID_OSProtectionShieldBroker;
v11[11] = CLSID_DataProtectionShieldBroker;
v11[12] = CLSID_ForceFieldWebProtectionShieldBroker;
v11[13] = CLSID_ApplicationGuardShieldBroker;
v11[14] = CLSID_ManagementShieldBroker;
```

Inspecting Cross-Session COM Objects

- ShieldProcessLauncherBroker Methods

```
ShieldProcessLauncherBroker::ShieldProcessLauncherBroker::RuntimeClassInitialize(void)
ShieldProcessLauncherBroker::ShieldProcessLauncherBroker::StartCleanPC(void)
ShieldProcessLauncherBroker::ShieldProcessLauncherBroker::TurnTimeServiceOn(void)
ShieldProcessLauncherBroker::ShieldProcessLauncherBroker::LaunchTroubleShooter(_tagDiagnosticTroubleShooter)
ShieldProcessLauncherBroker::ShieldProcessLauncherBroker::LaunchThirdPartyRemediationBinary(ushort *,SecurityProductType)
ShieldProcessLauncherBroker::ShieldProcessLauncherBroker::LaunchThirdPartyRemediationBinaryEx(ushort *,SecurityProductLaunchParar
ShieldProcessLauncherBroker::ShieldProcessLauncherBroker::LaunchAdvancedFirewallSettingsMMC(void)
ShieldProcessLauncherBroker::ShieldProcessLauncherBroker::LaunchTpmInitClear(void)
ShieldProcessLauncherBroker::ShieldProcessLauncherBroker::LaunchFolderInExplorer(ushort *)
ShieldProcessLauncherBroker::ShieldProcessLauncherBroker::LaunchOptionalFeaturesDialog(void)
```

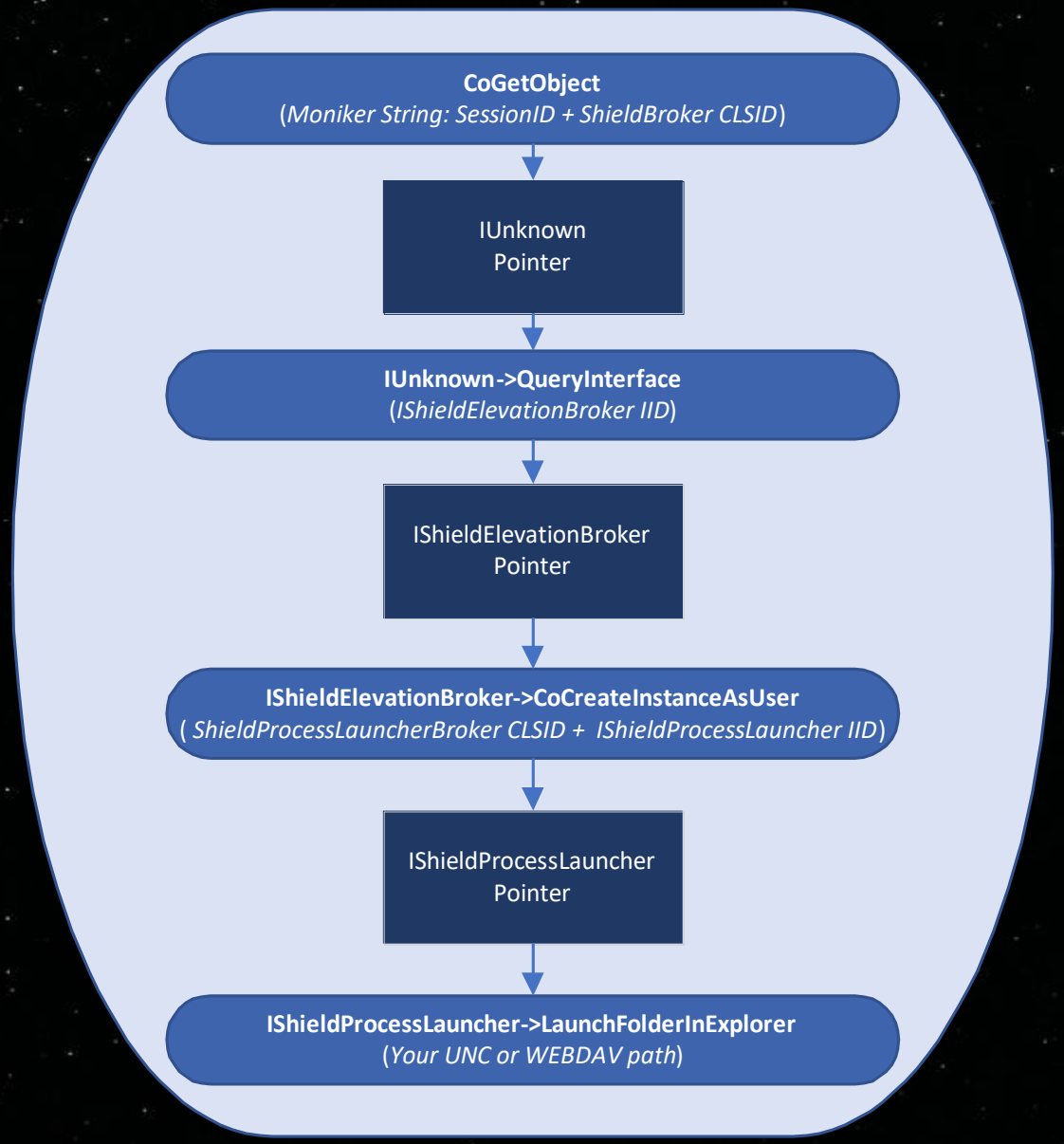
Inspecting Cross-Session COM Objects

- HardwareShieldBroker Methods

```
HardwareShieldBroker::EnableHvci(int *)
HardwareShieldBroker::DisableHvci(int *,int)
HardwareShieldBroker::HvciCancelIncompatibilityScan(void)
HardwareShieldBroker::HvciVelocityRollback(void)
HardwareShieldBroker::UpdateCurrentDriverPackage(ushort const *)
HardwareShieldBroker::AddIncompatibleDriver(ushort const *)
HardwareShieldBroker::GetIncompatibleDrivers(uint *,_incompatibleDriver * *)
HardwareShieldBroker::IsHvciRecommended(int *)
HardwareShieldBroker::IsHvciCapable(int *,uint *)
HardwareShieldBroker::IsHvciEnabled(int *)
HardwareShieldBroker::IsSecureBioEnabled(int *)
HardwareShieldBroker::IsSecureFaceAvailable(int *)
HardwareShieldBroker::IsSecureFingerprintAvailable(int *)
HardwareShieldBroker::IsCredentialGuardEnabled(int *)
HardwareShieldBroker::EnableSystemGuard(void)
HardwareShieldBroker::DisableSystemGuard(void)
```

Building a Test Harness

- OleViewDotNet allows us to:
  - Set Interface method arguments
  - Instantiate COM objects (in different sessions)
  - Export Interface definitions (in C++)
- Test harness allows use to construct more ‘complex’ clients (e.g. for trampolines)
- Program flow for our case study



Command Prompt

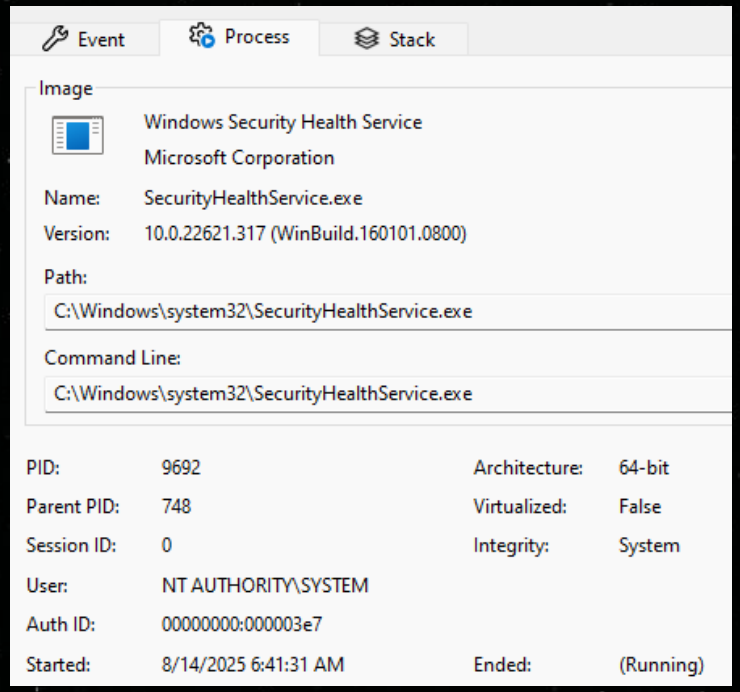
```
C:\Users\domainuser\Desktop>whoam
```

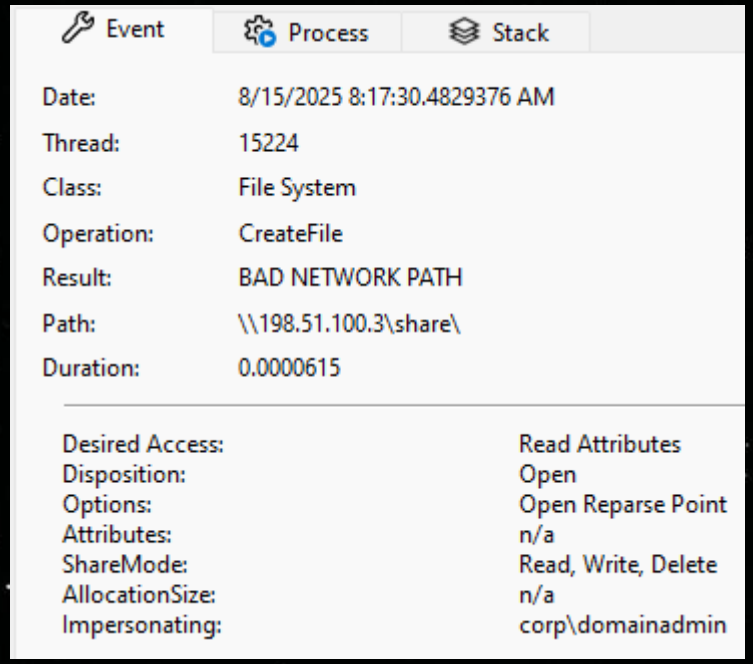
root@kali: ~

```
┌──(root㉿kali)-[~]
└─# impacket-ntlmrelayx -t smb://10.2.2.12 -smb2support -c "whoami"
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated compa
nies

[*] Protocol Client RPC loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server on port 445
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server on port 9389
[*] Setting up RAW Server on port 6666
[*] Multirelay disabled

[*] Servers started, waiting for connections
```

Case Study Results

- Root cause:
  - Impersonation mismatch in the Security Health Service
- MSRC reports resulted in two fixes:
  - **CVE-2025-53769**: Security Health Agent (ShieldProcessLauncherBrokerX)
  - **CVE-2025-47956**:Security Health Agent (HardwareShieldBrokerX)





Cross Session Post-Ex Tradecraft

There are
no solutions.

There are
only trade-offs.

– Thomas Sowell

Cross Session Post-Ex Tradecraft

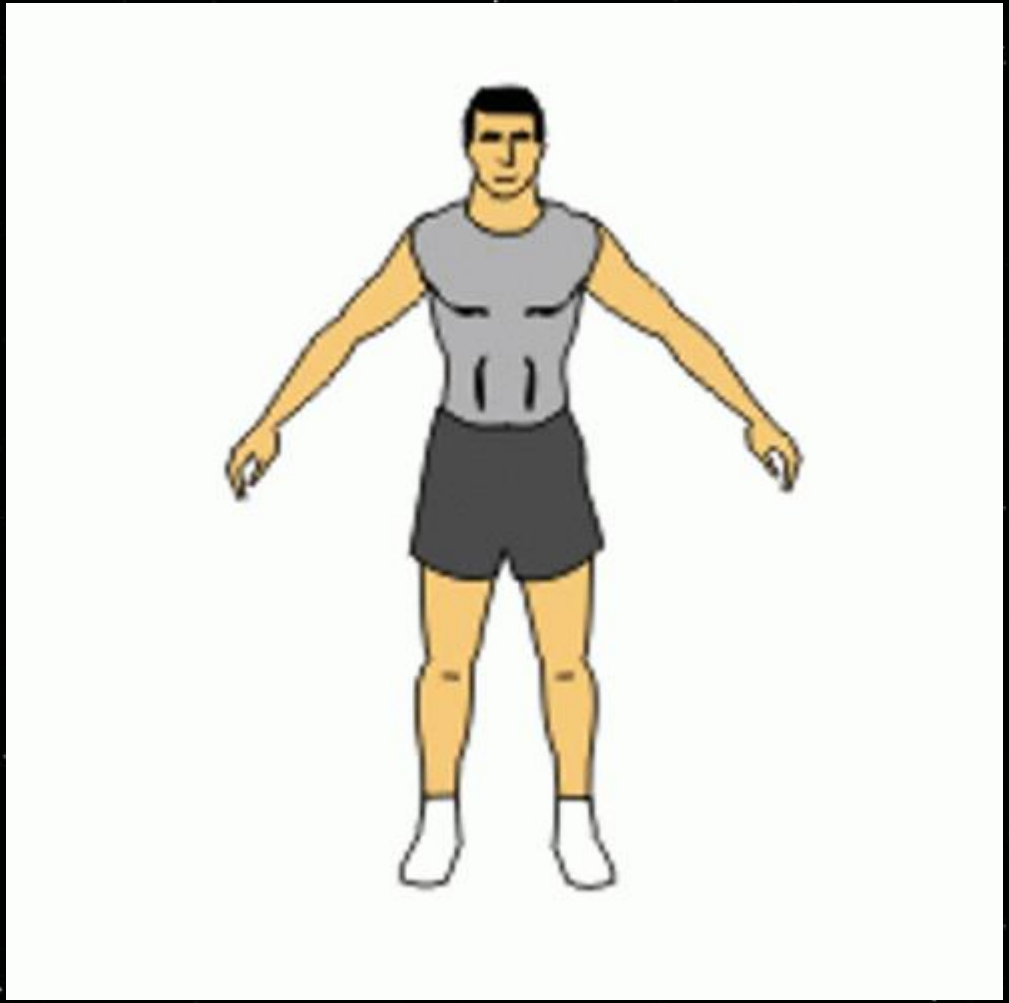
IHxExec

- Author: Michael Zhmailo
- Cross session command execution tool based on James Forshaw's CVE-2017-0010 EoP in the IHxHelpPaneServer COM class
- Executes in high-integrity context only
- Blog: https://cicada-8.medium.com/process-injection-is-dead-long-live-ihxhelppaneserver-af8f20431b5d
- Tool: https://github.com/CICADA8-Research/IHxExec

RemoteMonologue

- Author: Andrew Oliveau
- Windows credential harvesting technique that enables remote user compromise by leveraging the Interactive User RunAs key and coercing NTLM authentications via DCOM
- Blog: https://www.ibm.com/think/x-force/remotemonologue-weaponizing-dcom-ntlm-authentication-coercions
- Tool: https://github.com/xforcered/RemoteMonologue

DCOM Lateral Movement



Source: https://gifdb.com/images/high/demonstrating-lateral-raises-lbd5qrejay7c3qm1.gif

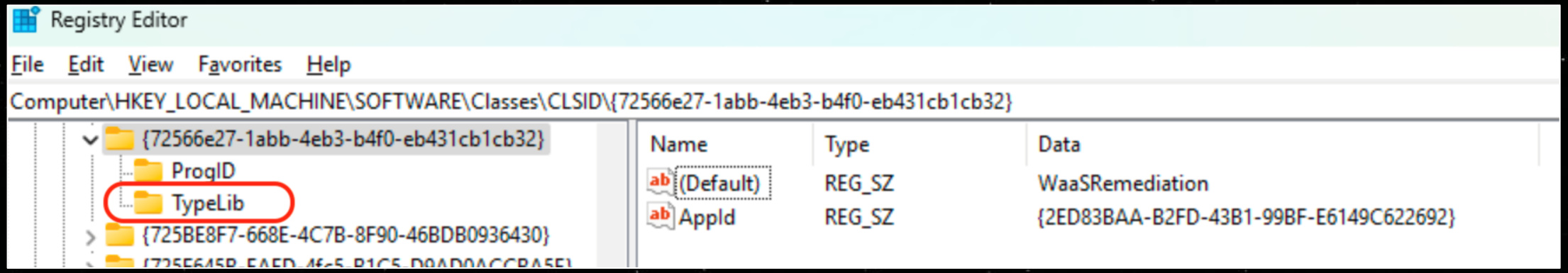
DCOM Lateral Movement – Current State

- Find DCOM Objects with interesting interface methods or access to other interfaces
  - MMC20.Application, ShellWindows, ShellBrowserWindow, MSIServer

- Apply DLL or COM hijacking to DCOM objects
  - https://github.com/WKL-Sec/dcomhijack
  - https://www.mdsec.co.uk/2020/10/i-live-to-move-it-windows-lateral-movement-part-3-dll-hijacking/
  - https://github.com/rtecCyberSec/BitlockMove

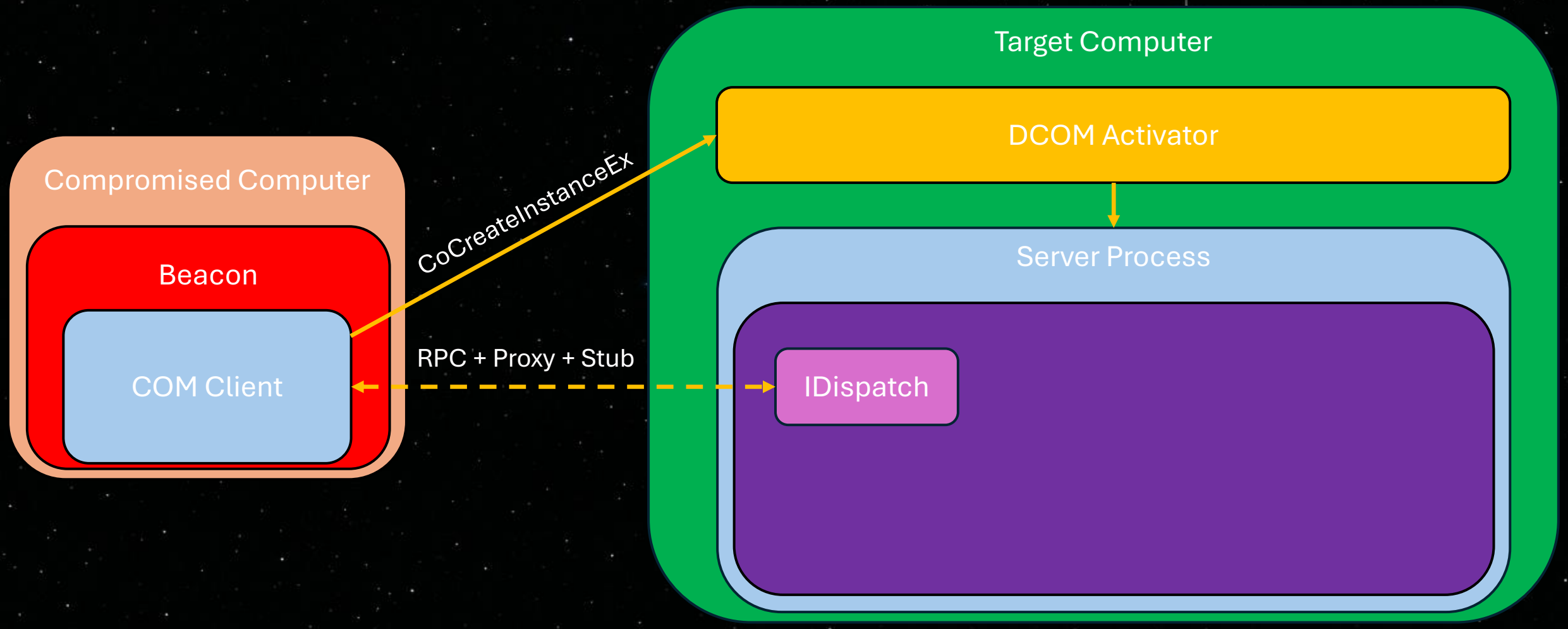
DCOM Lateral Movement – Trapped Objects

- James Forshaw introduces the Trapped COM Object bug class
- COM interfaces implementing IDispatch can utilize type libraries
  - Objects can be instantiated from those libraries and dependent libraries
  - Objects instantiated this way are instantiated in the server ("trapped")



- Used to obtain execution within separate PPL process (svchost.exe)
  - If it has an AppID, why not *DCOM* with it?

Trapped Objects – Instantiation



Trapped Objects – IDispatch::GetTypeInfo

IDispatch::GetTypeInfo method (oaidl.h)

02/22/2024

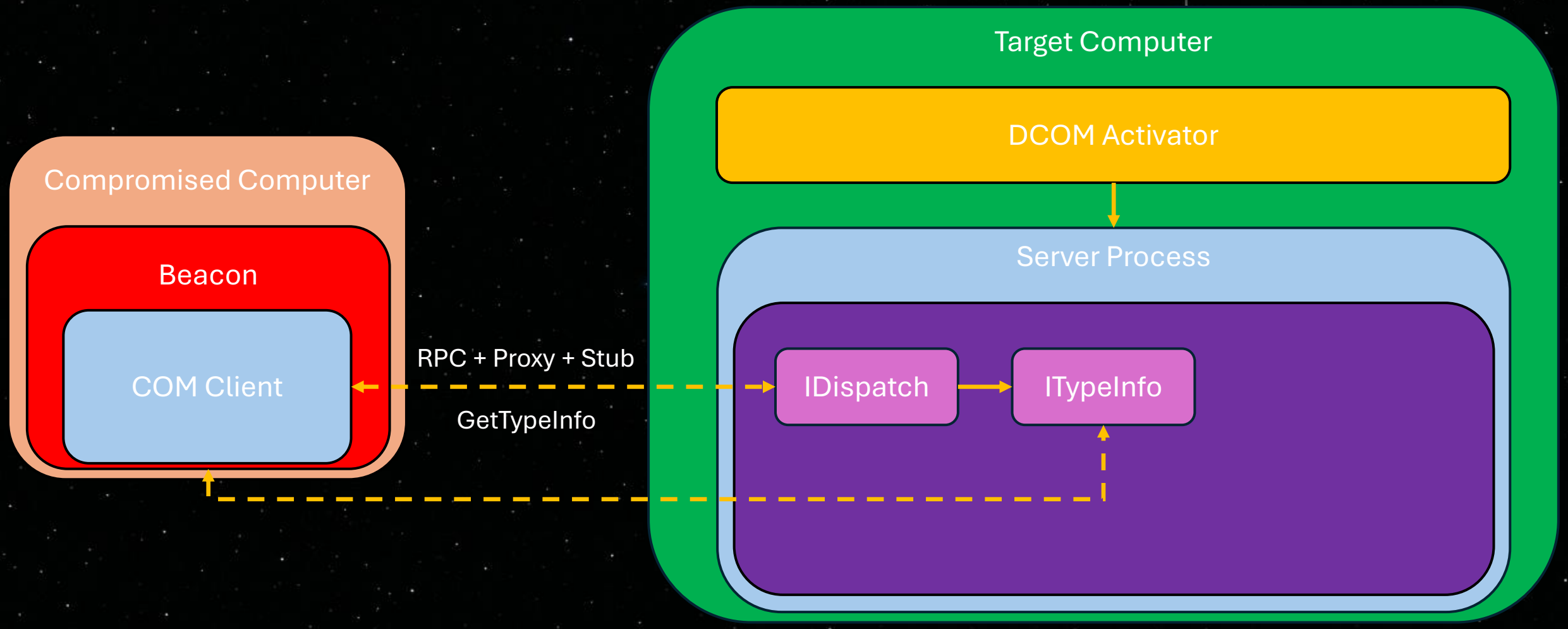
Retrieves the type information for an object, which can then be used to get the type information for an interface.

Syntax

C++ Copy

```
HRESULT GetTypeInfo(
  [in]  UINT      iTInfo,
  [in]  LCID      lcid,
  [out] ITypeInfo **ppTInfo
);
```

Trapped Objects – IDispatch::GetTypeInfo



Trapped Objects – ITypeInfo::GetContainingTypeLib

ITypeInfo::GetContainingTypeLib method (oaidl.h)

02/22/2024

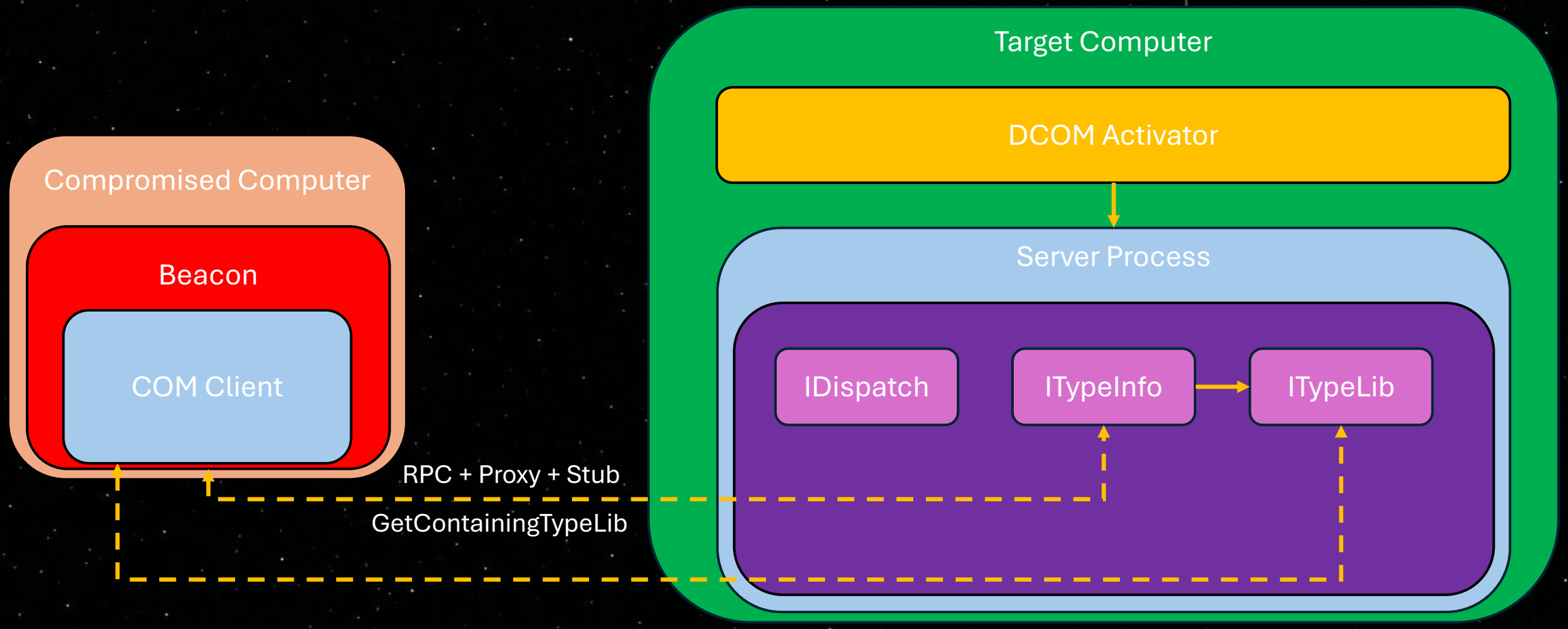
Retrieves the containing type library and the index of the type description within that type library.

Syntax

C++ Copy

```cpp
HRESULT GetContainingTypeLib(
  [out] ITypeLib **ppTLib,
  [out] UINT     *pIndex
);
```

Trapped Objects – ITypeInfo::GetContainingTypeLib



Trapped Objects – ITypeLib::GetTypeInfoOfGuid

ITypeLib::GetTypeInfoOfGuid method (oaidl.h)

02/22/2024

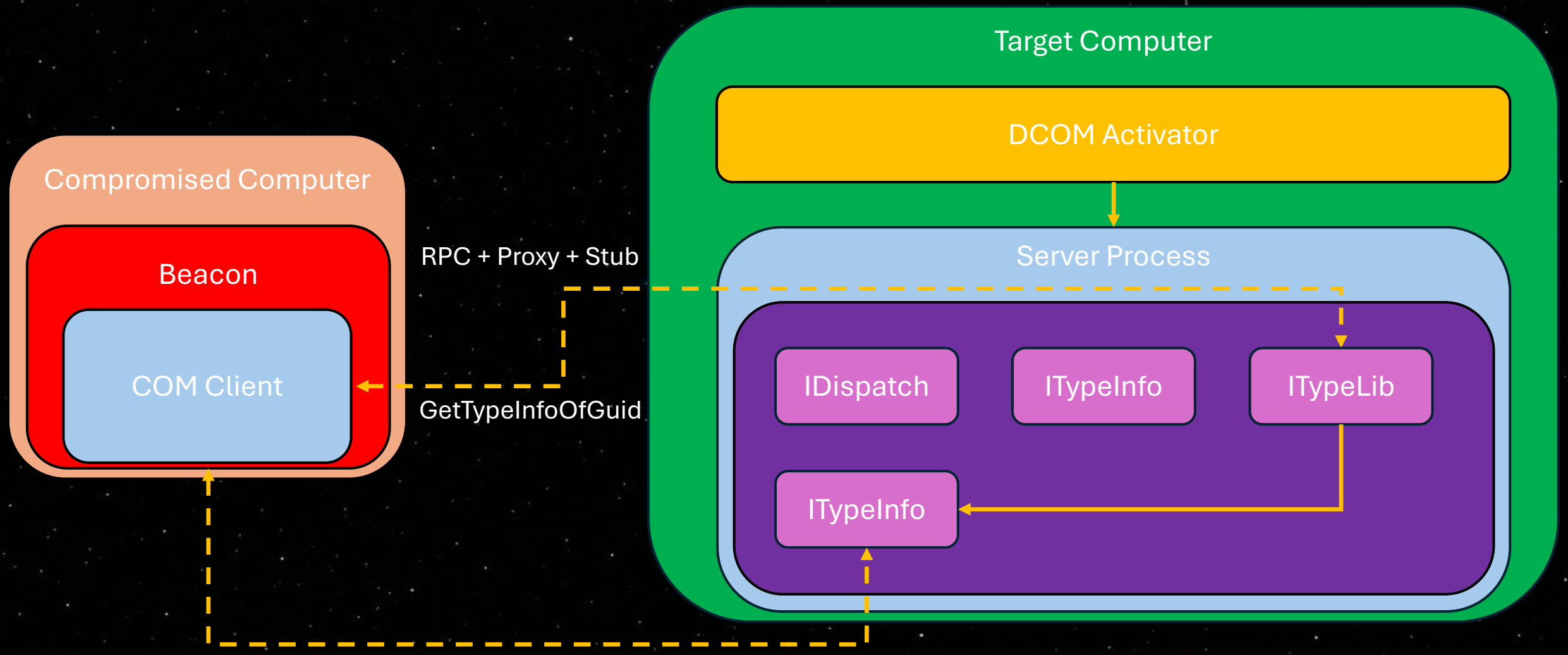
Retrieves the type description that corresponds to the specified GUID.

Syntax

C++ Copy

```
HRESULT GetTypeInfoOfGuid(
  [in]  REFGUID   guid,
  [out] ITypeInfo **ppTinfo
);
```

Trapped Objects – ITypeLib::GetTypeInfoOfGuid



Trapped Objects – ITypeInfo::CreateInstance

ITypeInfo::CreateInstance method (oaidl.h)

02/22/2024

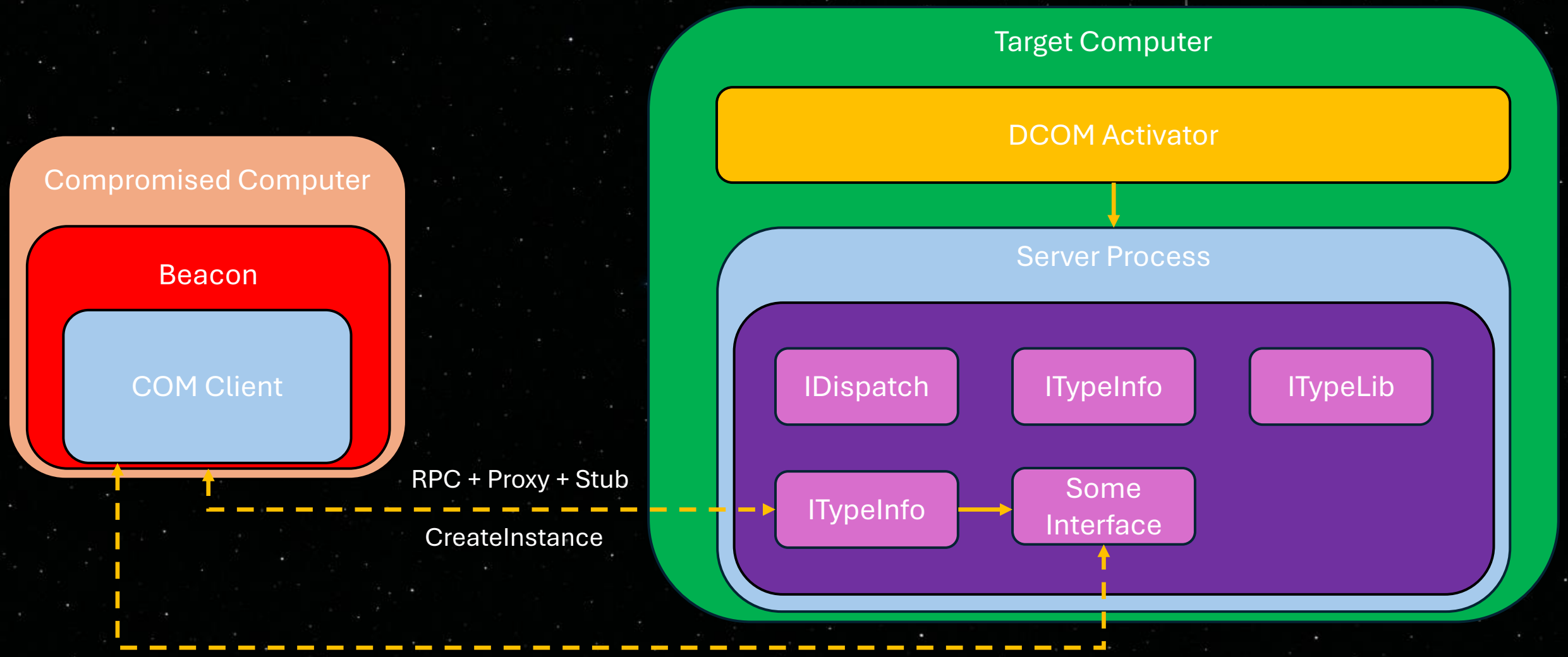
Creates a new instance of a type that describes a component object class (coclass).

Syntax

C++ Copy

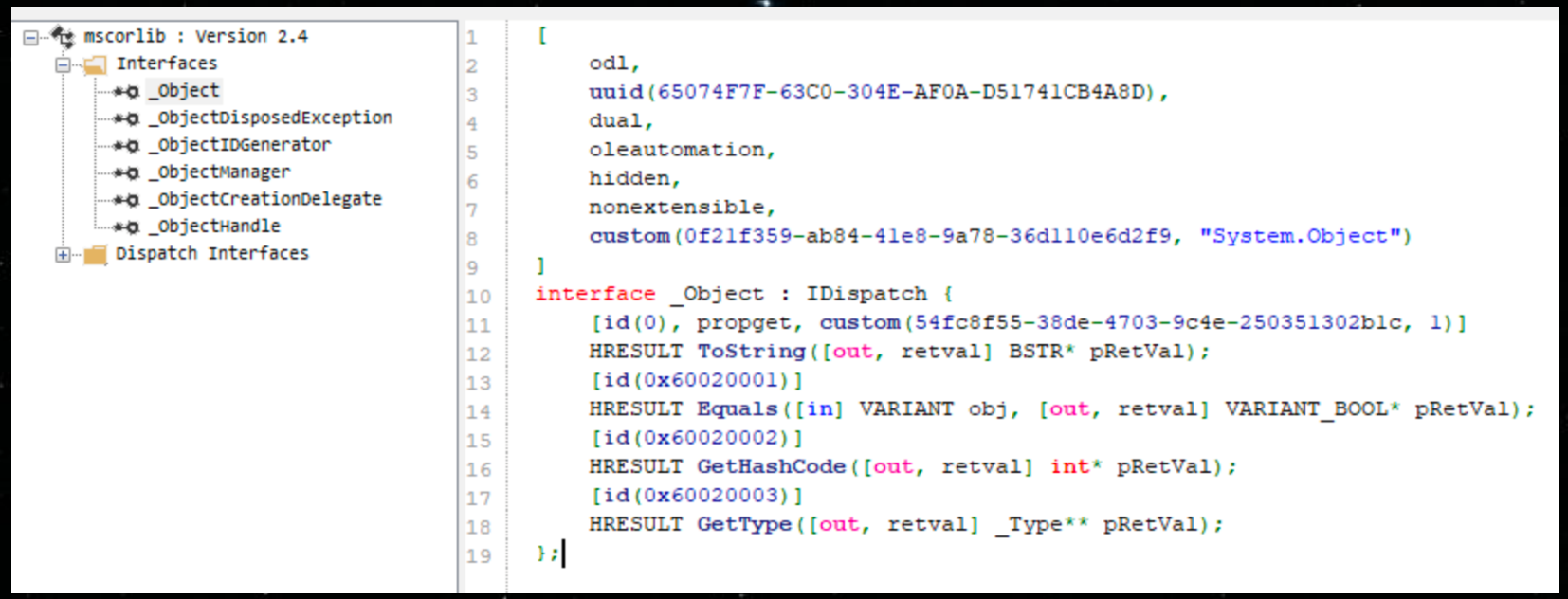
```cpp
HRESULT CreateInstance(
  [in]  IUnknown *pUnkOuter,
  [in]  REFIID   riid,
  [out] PVOID    *ppvObj
);
```

Trapped Objects – ITypeInfo::CreateInstance



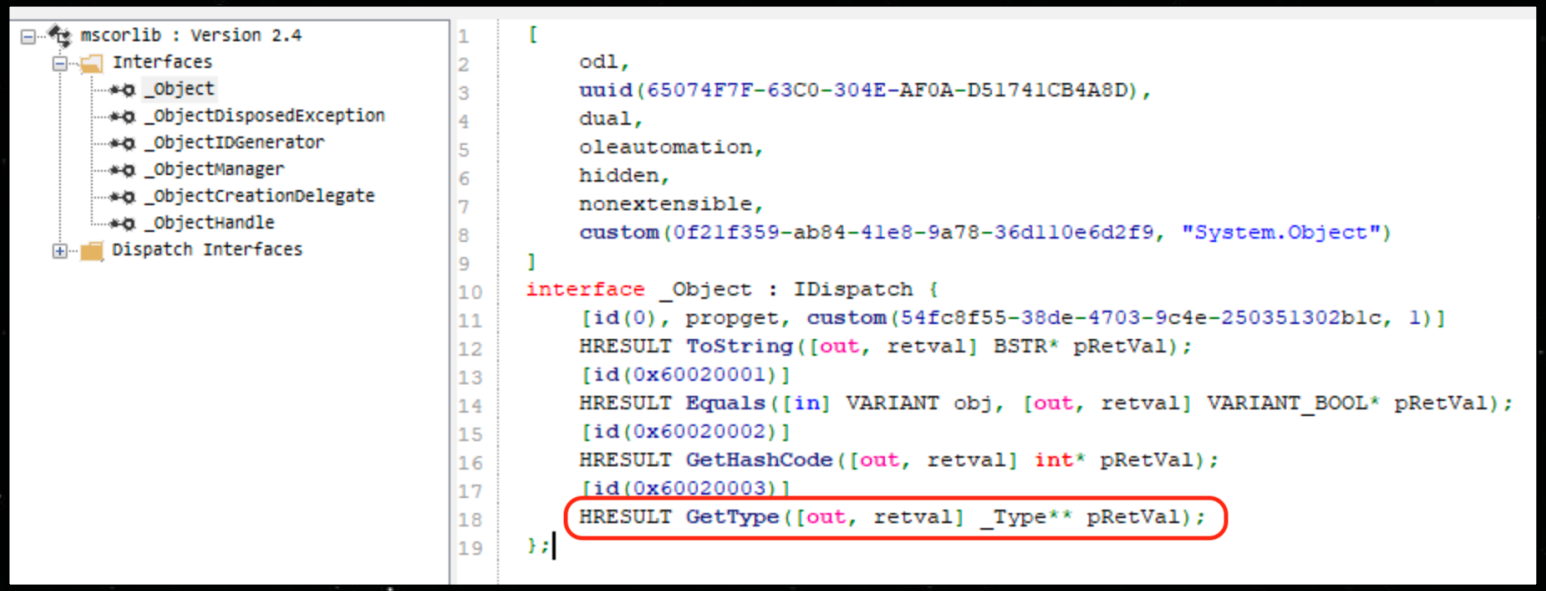
Trapped Objects – TreatAs and .Net Framework

- "Specifies the CLSID of a class that can emulate the current class." - https://learn.microsoft.com/en-us/windows/win32/com/treatas
- We can now instantiate *any* object server-side
  - What object would be interesting to us?
- .NET Objects are accessible and instantiable by COM



Trapped Objects – .Net Framework

- Assembly.Load is a static method; we are instantiating objects
- .NET Reflection can be done via System.Type (_Type) and some others
- .NET Reflection over DCOM requires **AllowDCOMReflection** value to be set



Select cmd.exe (running as sevenkingdoms\snaplabs)

```
c:\>wmic /node:kingslanding.sevenkingdoms.local process list brief | findstr /i cmd.exe
```

Trapped Objects: Review and Next Steps

- ForsHops was a POC, used WaasMedicSvc
  - Some changes in Windows 11 broke this (read the Forshaw blog)
- **Surely there are other objects with typelib + AppID + IDispatch**
- PPL is nice, but not necessary

Trapped Objects: Alternative Objects

- https://www.outflank.nl/blog/2025/07/29/accelerating-offensive-research-with-llm/
  - Kyle Avery harvested a dataset and had an agent build POCs
- PowerShell loop for CLSIDs with:
  - AppID
  - TypeLibrary subkey
  - Implement IDispatch

Source: https://www.doughusen.com/wp-content/uploads/2021/02/the-way-1024x576.jpg

```
PS C:\Users\nigel> Import-Module OleViewDotNet
PS C:\Users\nigel> # Define the base registry key for CLSIDs
PS C:\Users\nigel> $baseKey = "HKLM:\software\classes\CLSID"
PS C:\Users\nigel>
PS C:\Users\nigel> # Get all CLSID subkeys
PS C:\Users\nigel> $clsidKeys = Get-ChildItem -Path $baseKey
PS C:\Users\nigel>
PS C:\Users\nigel> foreach ($key in $clsidKeys) {
>>     try {
>>         # Check for the presence of InprocServer32 and TypeLib subkeys
>>         $inprocServer32Key = Get-Item -LiteralPath "$($key.PSPath)\InprocServer32" -ErrorAction SilentlyContinue
>>         $typeLibKey = Get-Item -LiteralPath "$($key.PSPath)\TypeLib" -ErrorAction SilentlyContinue
>> $localServer32Key = Get-Item -LiteralPath "$($key.PSPath)\LocalServer32" -ErrorAction SilentlyContinue
>>
>>         # Check for the AppID value in the CLSID key
>>         $appIDValue = Get-ItemProperty -Path $key.PSPath -Name "AppID" -ErrorAction SilentlyContinue
>>
>>         if ($appIDValue -and $typeLibKey) {
>>         $obj = New-ComObject -Clsid $key.PSChildName -iid "{00020400-0000-0000-C000-000000000046}"
>>             if ($localServer32Key) {
>>             Write-Output "CLSID: $($key.PSChildName), $($key.GetValue('')), $($localServer32Key.GetValue(''))"
>>             }
>>             else {
>>             Write-Output "CLSID: $($key.PSChildName), $($key.GetValue('')), dllhost.exe"
>>             }
>>         }
>>     } catch {
>>         # Handle errors silently
>>         continue
>>     }
>> }
CLSID: {0002123D-0000-0000-C000-000000000046}, Microsoft Publisher Application, C:\Program Files\Microsoft Office\Root\Office16\MSPUB.EXE /Automation
CLSID: {0002DF01-0000-0000-C000-000000000046}, Internet Explorer(Ver 1.0), "C:\Program Files\Internet Explorer\iexplore.exe"
CLSID: {0039FFEC-A022-4232-8274-6B34787BFC27}, Application Class, C:\Program Files\Microsoft Office\Root\Office16\ONENOTE.EXE
CLSID: {03837511-098B-11D8-9414-505054503030}, TraceDataProviderCollection, C:\WINDOWS\system32\plasrv.exe
CLSID: {03837513-098B-11D8-9414-505054503030}, TraceDataProvider, C:\WINDOWS\system32\plasrv.exe
CLSID: {0383751C-098B-11D8-9414-505054503030}, TraceSession, C:\WINDOWS\system32\plasrv.exe
CLSID: {03837521-098B-11D8-9414-505054503030}, DataCollectorSet, C:\WINDOWS\system32\plasrv.exe
CLSID: {03837525-098B-11D8-9414-505054503030}, DataCollectorSetCollection, C:\WINDOWS\system32\plasrv.exe
CLSID: {03837526-098B-11D8-9414-505054503030}, LegacyDataCollectorSet, C:\WINDOWS\system32\plasrv.exe
CLSID: {03837527-098B-11D8-9414-505054503030}, LegacyDataCollectorSetCollection, C:\WINDOWS\system32\plasrv.exe
CLSID: {03837528-098B-11D8-9414-505054503030}, LegacyTraceSession, C:\WINDOWS\system32\plasrv.exe
```

Trapped Objects: Modifying the PoC

```
//hr = CLSIDFromString(L"{72566E27-1ABB-4EB3-B4F0-EB431CB1CB32}", &clsid);    // WaaSRemediation
hr = CLSIDFromString(L"{03837513-098B-11D8-9414-505054503030}", &clsid);    // TraceDataProvider
//hr = CLSIDFromString(L"{00020820-0000-0000-C000-000000000046}", &clsid);    // Microsoft Excel 97-2003 Worksheet
//hr = CLSIDFromString(L"{0002123D-0000-0000-C000-000000000046}", &clsid);    // Microsoft Publisher Application
if (FAILED(hr)) {
    std::cout << "Invalid CLSID" << std::endl;
    CoUninitialize();
    return 1;
}
//hr = CLSIDFromString(L"{34050212-8AEB-416D-AB76-1E45521DB615}", &iid);    // IWaaSRemediation idk i picked randomly
hr = CLSIDFromString(L"{03837512-098B-11D8-9414-505054503030}", &iid);    // ITraceDataProvider
//hr = CLSIDFromString(L"{000208DA-0000-0000-C000-000000000046}", &iid);    // _Workbook
//hr = CLSIDFromString(L"{0002123e-0000-0000-c000-000000000046}", &iid);    // _Application
if (FAILED(hr)) {
    std::cout << "Invalid CLSID" << std::endl;
    CoUninitialize();
    return 1;
}
```

cmd.exe (running as sevenkingdoms\snaplabs)

```
c:\>wmic /node:kingslanding.sevenkingdoms.local process get Name,ParentProcessId,ProcessId | findstr /i "cmd.exe plasrv.exe"
```

Trapped Objects: Further Research

- Do we even need to escape to stdole?
  - Typelib Hijack and append additional CLSIDs?
  - This gets rid of the need TreatAs
- Alternatives to relying on .NET Reflection?
- Impacket (pain)

Defensive Considerations

General Recommendations

- Turn on the Windows Firewall
- Patch your systems (and don't forget 3rd party applications)
- Implement strong password policies
- Enforce idle disconnected session logout policies

Coercion/Relay/Harvesting Attacks

- Enable LDAP signing and channel binding
- Enforce SMB signing
- Detecting Registry manipulation (or creation) of suspicious RunAs Values: HKLM\SOFTWARE\Classes\APPID\{GUID}\RunAs

ForShops Lateral Movement Technique

- Detecting Registry manipulation (or creation) of suspicious TreatAs Keys: HKLM\SOFTWARE\Classes\CLSID\{GUID}\TreatAs
- Hunting for the presence of enabled OnlyUseLatestCLR and AllowDCOMReflection values in HKLM\SOFTWARE\Microsoft.NETFramework

COM
TO THE
DARKSIDE

THE END